# HOW TO BUILD A CI/CD PIPELINE USING JENKINS



An effective <u>continuous integration (CI) and continuous delivery (CD) pipeline</u> is essential for modern DevOps teams to cope with the rapidly evolving technology landscape. Combined with agile concepts, a fine CI/CD pipeline can streamline the software development life cycle resulting in higher-quality software with faster delivery.

In this article, I will discuss:

- What to know before building a CI/CD pipeline
- Jenkins CI/CD pipeline example
- Steps for setting up a Jenkins pipeline
- <u>Azure CI/CD pipeline example</u>
- And more!

(This article is part of our <u>DevOps Guide</u>. Use the right-hand menu to go deeper into individual practices and concepts.)

## What to know before building your CI/CD pipeline

#### What is a CI/CD pipeline?

The primary goal of a <u>CI/CD pipeline</u> is to automate the <u>software development lifecycle (SDLC)</u>.

The pipeline will cover many aspects of a software development process, from writing the code and

<u>running tests</u> to <u>delivery and deployment</u>. Simply stated, a CI/CD pipeline integrates automation and continuous monitoring into the development lifecycle. This kind of pipeline, which encompasses all the stages of the software development life cycle and connects each stage, is collectively called a CI/CD pipeline.

It will reduce manual tasks for the <u>development team</u> which in turn reduces the number of human errors while delivering fast results. All these contribute towards the increased productivity of the delivery team.

(Learn more about stages in a CI/CD pipeline, deployment pipelines and the role of CI/CD.)

## What is Jenkins?

Jenkins is an open source server for continuous integration and continuous deployment (CI/CD) which automates the build, test, and deploy phases of software development. With numerous plugins you can easily integrate, along with choices of tools, programming languages, and cloud environments, Jenkins is highly flexible and makes it easier to efficiently develop reliable apps.

### Why use Jenkins for CI/CD

You might wonder why Jenkins is a good option for building your CI/CD pipeline. Here are some of the reasons it is popular:

- Extensive plugin support: Whatever you want to do, there is likely already a plugin for it, which speeds and simplifies your work.
- Active open source community: Being free of licensing costs is just the beginning. It is supported by an active community, contributing solutions, advice, ideas, and tutorials.
- Platform independent: You are not tied to a specific operating system.
- **Scalable**: You can add nodes as needed and even run builds on different machines with different operating systems.
- Integratable: Whatever tools you are using, you can likely use them with Jenkins.
- Time-tested: Jenkins was one of the first CI/CD tools, so it is tried and true.

## Jenkins CI/CD pipeline example

What does a CI/CD pipeline built using Jenkins look like in action? Here is a simple web application development process.



Traditional CI/CD pipeline

- 1. The developer writes the code and commits the changes to a centralized code repository.
- 2. When the repo detects a change, it triggers the Jenkins server.
- 3. Jenkins gets the new code and carries out the automated build and testing. If any issues are detected while building or testing, Jenkins automatically informs the development team via a pre-configured method, like email or Slack.
- 4. The final package is uploaded to AWS Elastic Beanstalk, an application orchestration service, for production deployment.
- 5. Elastic Beanstalk manages the provisioning of infrastructure, <u>load balancing</u>, and scaling of the required resource type, such as EC2, RDS, or others.

The tools, processes, and complexity of a CI/CD pipeline vary from this example. Much depends on your development requirements and the business needs of your organization. Typical options include a straightforward, four-stage pipeline and a multi-stage concurrent pipeline — including multiple builds, different test stages (<u>smoke test, regression test</u>, user acceptance testing), and a multi-channel deployment (web, mobile).

## 8 steps to build a CI/CD pipeline using Jenkins

In this section, I'll show how to configure a simple CI/CD pipeline using Jenkins.

Before you start, make sure Jenkins is properly configured with the required dependencies. You'll also want a basic understanding of Jenkins concepts. In this example, Jenkins is configured in a Windows environment.

### **Step 1: Install Jenkins**

Download Jenkins from the official website and install it. Install Jenkins using the following Docker command:

```
docker run -d -p 8080:8080 jenkins/jenkins:lts
```



#### **Step 2: Configure Jenkins and add necessary plugins**

Configuring Jenkins is a matter of choosing the plugins you need. Git and Pipeline are commonly used tools you might want to add from the start.



### **Step 3: Open Jenkins**

Login to Jenkins and click on "New Item."

🏘 Jenkins	
Dashboard +	
🖀 New Item	
Reople New Item	
Build History	
Manage Jenkins	
Lockable Resources	
New View	
Build Queue	^
No builds in the queue.	
Build Executor Status	^
1 Idle	
2 Idle	
localhost:8080/view/all/newJob	

#### **Step 4: Name the pipeline**

Select the "Pipeline" option from the menu, provide a name for the pipeline, and click "OK."

🏟 Jenkins	Q search ⑦	L Admin	
Dashboard > All >			
	Enter an item name Test Pipeline * Required field		
	Freestyle project This is the central feature of Jenkins, Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.		
	Pipeline Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.		
	Multi-configuration project Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.		
	Folder Creates a container that stores nested items in it. Useful for grouping things together, Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.		
	GitHub Organization Scans a GitHub organization (or user account) for all repositories matching some defined markers.		
	GitHub Organization Scans a GitHub organization (or user account) for all repositories matching some defined markers.		
	Multibranch Pipeline Creates a set of Pipeline projects according to detected branches in one SCM repository.		
	ок		

#### **Step 5: Configure the pipeline**

We can configure the CI/CD pipeline in the pipeline configuration screen. There, we can set build triggers and other options for the pipeline. The most important section is the "Pipeline Definition" section, where you can define the stages of the pipeline. Pipeline supports both declarative and scripted syntaxes.

(Refer to the official Jenkins documentation for more detail.)

Let's use the sample "Hello World" pipeline script:

<pre>pipeline {     agent</pre>	any	
stages st } }	; { :age('Hello') { steps { echo 'Hello World' }	
Dashboard 🕐 Test Pipelin	Build Triggers     Advanced Project Options     Pipeline       O biasble this project     Image: Constraint of the project     Image: Constraint of the project       Quiet period     Image: Constraint of the project     Image: Constraint of the project       Trigger builds remotely (e.g. from scripts)     Image: Constraint of the project     Image: Constraint of the project	
	Advanced Project Options           Advanced         Advanced.           Pipeline         Image: Control of C	
	Definition	
	Pipeline script 0	1
	1 * pistine (       2 agent any       4 *       staget(vellor) (       5 *       7 *       1 *	
	Use Groovy Sandbox Pipeline Syntax	
	Save Apply	DET AR Inchine 2 Yes

Click on Apply and Save. You have configured a simple pipeline!

### **Step 6: Execute the pipeline**

Click on "Build Now" to execute the pipeline.

🏟 Jenkins		Q search ⑦	🚨 Admin	🛨 log out
Dashboard 🕐 Test Pipeline 🖻				
Back to Dashboard    Status  Changes  Build Now  Configure Build Now  Suid Now  Configure Build Now	Pipeline Test Pipeline			add description Disable Project
<ul> <li>Delete Pipeline</li> <li>Full Stage View</li> <li>Rename</li> <li>Pipeline Syntax</li> </ul>	No data available. This Pipeline has not yet run. Permalinks			
Build History trend   find  Atom.feed.for.al & Atom.feed.for.failures	9 9 9			
iccillor1000.00/Text Bodew/buildTelas-Sure			REST API	Jenkins 2.277.2

This will result in the pipeline stages getting executed and the result getting displayed in the "Stage View" section. We've only configured a single pipeline stage, as indicated here:

🧌 Jenkins		Q search 💿	Admin	🛨 log out
Dashboard 🕐 Test Pipeline 🔅				
Dashboard Test Pipeline   Back to Dashboard   Status   Changes   Changes   Duild Now   Configure   Delete Pipeline   Pull Stage View   Pipeline Syntax   Pipeline Syntax   Pipeline Syntax   Act IT. 2021. 600 AM   Act IT. 2021. 600 AM	Pipeline Test Pipeline			add description Diuble Project
			REST API	Jenkins 2.277.2

We can verify that the pipeline has been successfully executed by checking the console output for the build process.

🏟 Jenkins	Q search ⑦	🛓 Admin	🛨 log out
Dashboard > Test Pipeline > #1			
1 Back to Project	Console Output		
Q Status	Started by user Admin Remaing in Durability level: MAX_SUMVIVABILITY [Piesled] Start of Piesled		
Console Output	[Pipeline] node Running on Zenkins in C:\Users\bisin\AppDuta\Local\Jenkins\.jenkins\userkspace\Test Pipeline [Pipeline]		
View as plain text	[Pipeline] stage [Pipeline] ( (Mello) [Pipeline] echo		
Edit Build Information	Hells Horld [Pipeline] }		
S Delete build '#1'	[Pipeline] // stage [Pipeline] // node		
🔞 Restart from Stage	[Pigeline] End of Pigeline Finished: SUCCESS		
🕐 Replay			
Pipeline Steps			
Workspaces			
		REST API	Jenkins 2.277.2

#### **Step 7: Expand the pipeline definition**

Let's expand the pipeline by adding two more stages to the pipeline. For that, click on the "Configure" option and change the pipeline definition according to the following code block.

```
pipeline {
    agent any
    stages {
        stage('Stage #1') {
            steps {
                echo 'Hello World'
                sleep 10
                echo 'This is the First Stage'
            }
        }
        stage('Stage #2') {
            steps {
                echo 'This is the Second Stage'
            }
        }
        stage('Stage #3') {
            steps {
                echo 'This is the Third Stage'
            }
        }
    }
```

}

Save the changes and click on "Build Now" to execute the new pipeline. After successful execution, we can see each new stage in the Stage view.

🏘 Jenkins				
Dashboard > Test Pipeline >				
🔶 Back to Dashboard	Pipeline Test Pip	peline		
Status				
Changes				
Duild Now	Recent Changes			
🖑 Configure	Stage View			
S Delete Pipeline		Stage #1	Stage #2	Stage #3
G Full Stage View	Average stage times:	10s	32ms	36ms
Z Rename	(Average <u>full</u> run time: ~10s)			
O Pipeline Syntax	Apr 17 No 06:07 Changes	10s	32ms	36ms
Build History trend ^				
find X	Permalinks			
Apr. 17, 2021, 6:07, AM	<ul> <li>Last build (#1), 6 min 45 sec ag</li> <li>Last stable build (#1), 6 min 45</li> </ul>	o sec ago		
Apr 17. 2023, bold AM	<ul> <li>Last successful build (#1), 6 min</li> <li>Last completed build (#1), 6 min</li> </ul>	n 45 sec ago in 45 sec ago		
B) Atom need for all B) Atom need for failures				

The following console logs verify that the code was executed as expected:

襣 Jenkins		Q search	💄 Admin	🛨 log out
Dashboard > Test Pipeline > #2				
摿 Back to Project	Console Output			
G_ Status	Started by user Admin			
Changes	Running in Durability level: MAX_SUMVIVABILITY [Pipeling] Start of Pipeline [Pipeling] node			
Console Output	Rumning on Jenkins in C:\Users\bisin\AppData\Local\Jenkins\.jenkins\workspace\Test Pipeline [Pipeline] {			
View as plain text	[Pipeline] stage [Pipeline] ( Stage #1) [Pipeline] echo			
Edit Build Information	Hello World [Placing] Lieso			
O Delete build '#2'	Sleeping for 10 sec [Pipeline] echo			
Restart from Stage	This is the first Stage [Pipeline] } [Pipeline] // stage			
🕐 Replay	[Pipeline] stage [Pipeline] ( (Stage #2)			
Pipeline Steps	[Pipeline] echo This is the Second Stage			
Workspaces	[Pipeline] ) [Pipeline] // stage [Pineline] ctame			
Previous Build	<pre>[Pipeline] toge [Pipeline] (Stage #3) [Pipeline] etho [Pipeline] // stage [Pipeline] // stage [Pipeline] // node [Pipeline] // node [Pipeline] tod of Pipeline Finished: SUCCESS</pre>			
			REST API	Jenkins 2.277.2

#### **Step 8: Visualize the pipeline**

We can use the "<u>Pipeline timeline</u>" plugin for better visualization of pipeline stages. Simply install the plugin, and inside the build stage, you will find an option called "Build timeline."



Click on that option, and you will be presented with a timeline of the pipeline events, as shown below.

Build timeline > #2										
Longest sta	ge: Stage #1 (10 seconds)				Ended on April 17th 202	21, 06:07:28 am				
Stage #1	Sleep									
Stage #2										Print Message
Stage #3										Print Message
		19	20	21 2	2	23	24	25	26	27 28

That's it! You've successfully configured a CI/CD pipeline in Jenkins.

The next step is to expand the pipeline by integrating:

- External code repositories
- Test frameworks
- Deployment strategies

Good luck!

## **Cloud-based Azure CI/CD pipeline example**

With the increased adoption of <u>cloud technologies</u>, the growing trend is to move the DevOps tasks to the cloud. Cloud service providers like Azure and AWS provide a full suite of services to manage all the required DevOps tasks using their respective platforms.

The following is a simple cloud-based DevOps CI/CD pipeline entirely based on <u>Azure (Azure</u> <u>DevOps Services) tools.</u>



Cloud-based CI/CD pipeline using Azure

- 1. A developer changes existing or creates new source code, then commits the changes to Azure Repos.
- 2. These repo changes trigger the Azure Pipeline.
- 3. With the combination of Azure Test Plans, Azure Pipelines builds and tests the new code changes. (This is the Continuous Integration process.)
- 4. Azure Pipelines then triggers the deployment of successfully tested and built artifacts to the required environments with the necessary dependencies and environmental variables. (This is the Continuous Deployment process.)
- 5. Artifacts are stored in the Azure Artifacts service, which acts as a universal repository.
- 6. Azure application monitoring services provide the developers with real-time insights into the deployed application, such as health reports and usage information.

In addition to the CI/CD pipeline, Azure also enables managing the SDLC using Azure Boards as an agile planning tool. Here, you'll have two options:

• Manually configure a complete CI/CD pipeline

• Choose a SaaS solution like Azure DevOps or DevOps Tooling by AWS

### **CI/CD** pipelines minimize manual work

A properly configured pipeline will increase the productivity of the delivery team by reducing the manual workload and eliminating most manual errors while increasing the overall product quality. This will ultimately lead to a faster and more agile development life cycle that benefits end-users, developers, and the business as a whole.

Learn from the choices Humana made when selecting a modern mainframe development environment for editing and debugging code to improve their velocity, quality and efficiency.

## **Related reading**

- <u>BMC DevOps Blog</u>
- SRE vs DevOps: What's The Difference?
- How Containers Fit in a DevOps Delivery Pipeline
- How & Why To Become a Software Factory
- Book Review of The Phoenix Project: DevOps For Everyone
- Enterprise DevOps: Increase Agility Across the Enterprise